A. Problem Statement: The goal of this problem is to find out the time it takes to go through a linked list. First by inserting and sorting at the same time, a bunch of Integers from a text document until the last integer in the document has been added. The next part of the problem has you find the min , man and median of the linked list. Each one of these have to be timed as well, checking how long it takes to find each value. The time should be in either seconds, milliseconds, or microseconds, which ever is the most accurate for each value.

B. Algorithm design decisions: So starting with the min and max functions. These really are straight forward. Seeing as the linked list is to already be sorted before calling these functions, the time it takes to simply go into the list and get the values is quick. Min is just the first value in the linked list, IE at the element 0. While, Max is at the “sizeof” or the amount of elements in the linked list minus one to account for the way linked lists are numbers. The size function for Linked list already does such a thing so there is no need for the minus 1. However, there are 2 ways to get min and max, that require a more algebraic approach. These ways include getting the median, which I see no other way but to go through the entire Linked list and add up all the values until the end of the list. After that is done just divide it by size (+1 if you use the Linked list size option) to get the median.

   Now to get the 2 other ways for min and max. This can be done at the same time as median, meaning, while you are adding all the values for the median you can check each value to make sure that the min and max are the correct values. The least useful way to do this would be to do it separately on each function, meaning to run a whole for loop going through all the elements of the linked list 3 times for the min max and median functions thereby eating up a lot of wasted time.

   I always had my own way for inserting things into the linked list. Reason being, I could not find an insert for the Linked list class. I just set it up with some easy to understand, saying that if the first element is empty you put the first integer there. If the second integer is lower then the first element you place it in front of the linked list, but if it is higher or equal to the number you place it at the end. Following the placement of the second element, all additional elements will check if it is less than or equal to the first element, if so you place it in the front. If it is greater or equal to the last element, you place it in the back. If neither of these are placed you get the middle of the linked list using size divided by 2, and check if the value to be placed is first equal to that element's value. If so place it there, If not check if its lower, and higher. If it is lower to move towards the front of the list till you find where the value needs to be placed. If it is higher you do the same thing but towards the end of the list. I wanted to make this system better, probably making median be solved at the same time to help with the time it takes to traverse that as well.

C. Experimental Setup:

   Machine specification: - Cyberpower machine, Intel(R) Core(™) i7-8700k CPU @3.70GHz, 16 GB RAM, Dual harddrive: soildstate and standard(2TB drive)

How many times did I repeat: Over 50 for each one aside from min and max, which was lower at 20 each, I tested each element over and over again. I only used what information I had from previous classes. I also tested these elements with 3 other amounts of random text files.

O/S: Windows 10, MY JDK is 1.8.0_141, I wrote the code on notepad++ and geany, tested and ran through console commands, netbeans, and VS

D. Experimental Results & Discussions:

Input2.txt
Placed in Linked List time: 194, 191,213,189,201,191,217,202,
229,275,357,189,234,196,255,227,524,293,223,361,275,471,188,384,279,215,251

Median time: 64,121,26,27,27,28,43,27,29,28,
30,27,27,30,133,74,97,27,79,27,28,129,50,85,96,74,50

Min : 4,5,5,9,6,31,5,4,9,6,5,4,4,4,8,5,7,8,9,7,5,9,4,26,4,5,4
Max : 4,4,32,5,5,4,7,4,24,28,6,4,4,4,21,5,4,18,9,6,5,19,8,4,8,19,4,4

input.txt
Placed in Linked List time: 157780,165148,156866,152959,158752,153685,
153056,155514,158415,157620,155527, 152982,156188,154836 ,156630,
150893,159634,165784,159294,161842,151405

Median time:1511,1489,1400,2467,1705,1607 , 2118, 1583 ,2161 ,1447,1949
,1777,1512,1747,2052, 1870,1559, 2167, 1453,1722,1683

Min :4,3,1,1,1,0,0,1,15 , 0 ,1, 1, 0 ,1, 0, 0 , 0 ,1,2,2,1
Max :1,1,0,1,0,1,0,0,0 ,0 , 0, 1,0,1,0,1, 0,0,0 ,1,0

Discussion: Strange, When I look at these random tests that I have done, it looks like getting the max and min for the larger text file is quicker. Even when done in the middle of the running tests for smaller text files. I have no real idea why this is doing in such a way. The system does not run differently when having that. Perhaps the system has access to the memory easier? I really have no idea on that. The Median time and Linked List insertion time make far more sense. Seeing as these things have to go through the whole linked list to both count the median. It will always be the 2nd largest number for times on this list. Insertion will always be the largest time, since it needs to do multiple checks to see where it goes into it. Now, for min and max, the smaller text file will have min be faster then max, but that is because it has to only get the first linked list, while max has to get the larged. While, for a strange reason for the larger text file the min is slower then the max, granted it is really small, but I guess the only assumption I can make of it is that max is faster then min. There must be a reason why this is done.